CLIPPEDIMAGE= JP409330230A
PAT-NO: JP409330230A
DOCUMENT-IDENTIFIER: JP 09330230 A
TITLE: METHOD AND SYSTEM FOR TRANSPLANTING APPLICATION BINARY INTERFACE

PUBN-DATE: December 22, 1997

INVENTOR-INFORMATION:
NAME
SCOTT, HARRISON DANFORCE
PURAKASHIYU, BUINODORAI DESAI

ASSIGNEE-INFORMATION:
NAME COUNTRY
INTERNATL BUSINESS MACH CORP <IBM> N/A

APPL-NO: JP09054658
APPL-DATE: March 10, 1997

INT-CL_(IPC): G06F009/45

ABSTRACT:
PROBLEM TO BE SOLVED: To provide an information processing system equipped with
an application programming interface(API) with which an application binary interface(ABI) can be transplanted from a certain platform to another non-compatible platform almost without difficulty.

SOLUTION: Inside the information processing system, there are one or plural processors, storage system, one or plural input/output controllers, system bus for connecting the processors, storage system and input/output controllers, and operating system programmed to control the operation of the information processing system. The API is operated while being interlocked with an operating program. The API provides an interface neutral for languages and platforms for operating a specified parameter list inside the ABI and can be

transplanted from a certain platform to the other platform.

(19)日本国特許庁（ＪＰ） (12)公開特許公報（Ａ） (11)特許出願公開番号
特開平9－330230
(43)公開日 平成９年(1997)12月22日

(51)Int.Cl.⁶	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ 9/45			Ｇ０６Ｆ 9/44	３２０Ａ	

審査請求 未請求 請求項の数21 ＯＬ （全 10 頁）

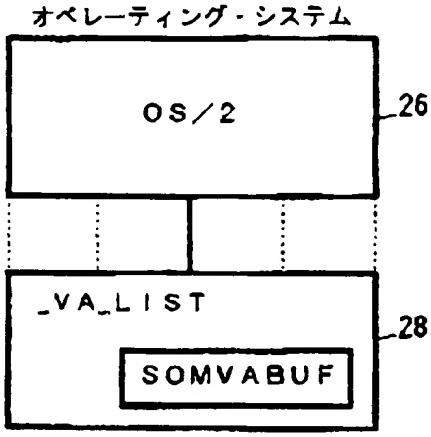
(21)出願番号 特願平9－54658
(22)出願日 平成９年(1997)３月10日
(31)優先権主張番号 ０８／６１９０６１
(32)優先日 1996年３月20日
(33)優先権主張国 米国（ＵＳ）

(71)出願人 390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州 アーモンク （番地なし）
(72)発明者 スコット・ハリソン・ダンフォース
アメリカ合衆国78750 テキサス州オースチン ウッドランド・ビレッジ 10011
(74)代理人 弁理士 合田 潔 （外２名）

最終頁に続く

(54)【発明の名称】 アプリケーション・バイナリ・インターフェースを移植する方法及びシステム

(57)【要約】
【課題】 アプリケーション・バイナリ・インターフェースを、あるプラットフォームから別の非互換プラットフォームにほとんど困難なしに移植することができるアプリケーション・プログラミング・インターフェース（ＡＰＩ）を含む情報処理システムを提供する。
【解決手段】 情報処理システム内には、１つまたは複数のプロセッサ、記憶システム、１つまたは複数の入出力制御装置、プロセッサと記憶システムと入出力制御装置を接続するシステム・バス、及び情報処理システムの動作を制御するようにプログラムされたオペレーティング・システムがある。ＡＰＩは、オペレーティング・プログラムと連動して動作する。ＡＰＩは、ＡＢＩ内の特定の変数リストを操作するための言語及びプラットフォームに中立なインターフェースを提供し、あるプラットフォームから別のプラットフォームに移植可能である。



1

【特許請求の範囲】

【請求項１】１つまたは複数のプロセッサと、
記憶システムと、
１つまたは複数の入出力制御部と、
プロセッサ、記憶システム及び入出力制御装置を接続するシステム・バスと、
情報処理システムの動作を制御するようにプログラムされたオペレーティング・システムと、
オペレーティング・システム・プログラムと連動して動作し、変数リストを操作するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェース（ＡＰＩ）プログラムとを含む情報処理システム。

【請求項２】アプリケーション・プログラミング・インターフェース（ＡＰＩ）プログラムが、さらに、
ユーザの可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する手段を含むことを特徴とする請求項１に記載のシステム。

【請求項３】ＡＰＩプログラムがさらに、
前記オブジェクト変数バッファ構造の１つに含まれる変数リストに、渡された引数を加える手段を含むことを特徴とする請求項２に記載のシステム。

【請求項４】ＡＰＩプログラムがさらに、
前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する手段を含むことを特徴とする請求項３に記載のシステム。

【請求項５】ＡＰＩプログラムがさらに、
前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する手段を含むことを特徴とする請求項３に記載のシステム。

【請求項６】前記ＡＰＩがさらに、
前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第１スカラ値を取り出す手段を含むことを特徴とする請求項３に記載のシステム。

【請求項７】前記ＡＰＩがさらに、
ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに、第１引数を入れることを可能にする手段を含むことを特徴とする請求項３に記載のシステム。

【請求項８】変数リストを操作して、アプリケーション・バイナリ・インターフェースを所与のプラットフォームから１つまたは複数の追加プラットフォームに移植するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェース（ＡＰＩ）を提供する手段を含むコンピュータ読取り媒体。

【請求項９】アプリケーション・プログラミング・インターフェースがさらに、
ユーザ可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する手段を含むことを特徴とする請求項８に記載のコンピュータ読取り媒体。

2

【請求項１０】ＡＰＩ手段がさらに、
前記オブジェクト変数バッファ構造の１つに含まれる変数リストに、渡された引数を追加する手段を含むことを特徴とする請求項９に記載のコンピュータ読取り媒体。

【請求項１１】ＡＰＩ手段がさらに、
前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する手段を含むことを特徴とする請求項９に記載のコンピュータ読取り媒体。

【請求項１２】ＡＰＩ手段がさらに、
前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する手段を含むことを特徴とする請求項８に記載のコンピュータ読取り媒体。

【請求項１３】ＡＰＩ手段がさらに、
前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第１スカラ値を取り出す手段を含むことを特徴とする請求項８に記載のコンピュータ読取り媒体。

【請求項１４】ＡＰＩ手段がさらに、
ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに、第１引数を入れることを可能にする手段を含むことを特徴とする請求項８に記載のコンピュータ読取り媒体。

【請求項１５】アプリケーション・バイナリ・インターフェース（ＡＢＩ）を、情報処理システム上で動作する選択されたプラットフォームから１つまたは複数の追加プラットフォームに移植する方法であって、
変数リストを操作して、前記ＡＢＩを前記選択されたプラットフォームから前記１つまたは複数の追加プラットフォームに移植するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェースを提供する段階を含む方法。

【請求項１６】前記アプリケーション・プログラミング・インターフェース（ＡＰＩ）がさらに、
ユーザ可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する段階を含むことを特徴とする請求項１５に記載のＡＢＩを移植する方法。

【請求項１７】前記ＡＰＩがさらに、
前記オブジェクト変数バッファ構造の１つに含まれる変数リストに、渡された引数を追加する段階を含むことを特徴とする請求項１６に記載のＡＢＩを移植する方法。

【請求項１８】前記ＡＰＩがさらに、
前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する段階

を含むことを特徴とする請求項16に記載のABIを移植する方法。

【請求項19】前記APIがさらに、

前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する段階を含むことを特徴とする請求項16に記載のABIを移植する方法。

【請求項20】前記APIがさらに、

前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラ値を取り出す段階を含むことを特徴とする請求項16に記載のABIを移植する方法。

【請求項21】前記APIがさらに、

ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リスト内に最初の引数を入れることを可能にする段階を含むことを特徴とする請求項16に記載のABIを移植する方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般に複数のプラットフォームを支援する情報処理システムに関する。より具体的には、本発明は、処理システム内で動作するプラットフォーム間で言語及びプラットフォームに中立なインターフェース処理を提供するアプリケーション・プログラミング・インターフェースに関する。

【0002】

【従来の技術】電子計算システムは、仕事や私生活のほとんどあらゆる側面に浸透している。コンピュータ・システムは、仕事面では、大量のデータの管理やそのようなデータの処理に使用される。このようなデータ処理が必要なため、様々なコンピュータ・システム上で動作するいくつかのプラットフォームが提供されている。このようなプラットフォームは、通常、各社独自のものであり、多くの場合ソフトウェア変換のために相互に対話するのが困難である。それが重要なのは、あるプラットフォーム上で強い人気をもつ特定のソフトウェア・プログラムを別のプラットフォームに移植するとき、プラットフォームの言語に互換性がないために、そのような人気のあるソフトウェアを新しいプラットフォームに簡単に移植することができないからである。

【0003】それぞれ各プラットフォームは、固有のアプリケーション・バイナリ・インターフェース（ABI）を有し、それが、メモリ内に可変引数がどのように記憶されているかを決定する。C言語では、Cライブラリ内に、可変数の引数を入力として受け取ると予想される機能のスタック・フレーム上に可変引数リストを作成するインターフェースが提供される。この場合、残念ながら、可変引数リストの作成を必要とするソフトウェア開発業者は、リンケージ規約の詳細と可変引数が所与のプラットフォームに対してどのように表されるかを知らなければならない。様々なプラットフォームが異なるABIを有する場合、複数プラットフォーム用の移植可能コードを記述するタスクはほとんど不可能になる。

【0004】

【発明が解決しようとする課題】したがって、アプリケーションを複数のプラットフォーム間で大した困難なしに移植できるようにするAPIが必要である。

【0005】

【課題を解決するための手段】本発明によれば、アプリケーション・バイナリ・インターフェースをあるプラットフォームから別の非互換プラットフォームにほとんど困難なしに移植できるように改良されたアプリケーション・プログラミング・インターフェースを含む情報処理システムが開示される。情報処理システム内には、1つまたは複数のプロセッサ、記憶システム、1つまたは複数の入出力制御装置、プロセッサと記憶システムと入出力制御装置を接続するシステム・バス、及び情報処理システムの動作を制御するようにプログラムされたオペレーティング・システムがある。アプリケーション・プログラミング・インターフェース（API）は、オペレーティング・プログラムと連動して動作する。APIは、ABI内の特定の変数リストを操作するための言語及びプラットフォームに中立なインターフェースを提供し、あるプラットフォームから別のプラットフォームに移植可能である。

【0006】アプリケーション・プログラミング・インターフェースは、さらに、選択されたオブジェクト変数バッファ構造を作成し初期設定する手段を含む。オブジェクト変数バッファ構造は、ユーザ可変引数を含むこともある。さらに、APIは、オブジェクト変数バッファ構造の1つに含まれる変数リストに渡された引数を追加する手段を含む。APIは、また、オブジェクト変数バッファ構造内の変数リストを入力引数として渡された変数リストに複写する手段も含む。さらに、APIは、オブジェクト変数バッファ構造と、オブジェクト変数バッファ構造内に含まれる変数リストとに関連付けられた任意の割り振りメモリを消去する手段を含む。さらに、APIは、オブジェクト変数バッファ構造内の初期設定済みの変数リストに記憶された第1スカラ値を取り出す手段を含む。また、APIはさらに、ユーザが、オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに最初の引数を入れることを可能にする手段を含む。APIは、情報処理システム内にロードされたプログラムまたはコンピュータ読取り媒体上に記憶されたプログラムとして実施され、あるいはプログラムを特定のプラットフォームから第2の非互換プラットフォームに移植する方法として実施されることが開示される。

【0007】

【発明の実施の形態】次に、図1と図2を参照し、本発明の情報処理システム10及びオペレーティング・シス

5

テム環境について説明する。

【０００８】情報処理システム１０は、高性能プロセッサ１２、記憶システム１４、システム・バス１６、表示装置２０を制御する表示サブシステム１８、カーソル制御装置２２、及び入出力制御装置２４を備え、これらの装置がすべてシステム・バス１６で接続されたグラフィック・ワークステーションなどである。図１に示したような情報処理システム１０は、ＯＳ／２（ＯＳ／２は、インターナショナル・ビジネス・マシーンズ・コーポレイションの登録商標）など市販の周知のマルチタスク・オペレーティング・システム２６（図２に示す）によって動作する。情報処理システム１０を動作させる際に、ＯＳ／２オペレーティング・システムが制御するタスクの１つは、特に、インターナショナル・ビジネス・マシーンズ・コーポレイションから市販の製品の一部であるＳＯＭｏｂｊｅｃｔｓに含まれるＳＯＭオブジェクトを使用するプログラムの実行である。

【０００９】本発明の方法及び製造物は、図２のブロック２８のＳＯＭｏｂｊｅｃｔプログラムに組み込むことができる。

【００１０】前述のように、所与のプラットフォーム用のアプリケーション・バイナリ・インターフェース（ＡＢＩ）は、本質的に各社独自のものであり、複数のプラットフォームに簡単に移植することはできない。この問題に対処するために、本発明は、変数リスト（va_list）を操作するための、言語及びプラットフォームに中立なインターフェースを提供する新しいアプリケーション・プログラミング・インターフェース（ＡＰＩ）を提供する。この改良型のＡＰＩは、ライブラリ内にパッケージ化された少なくとも６つの機能を提供する。これらの機能は、可変引数リストのユーザからＡＢＩの低レベルの詳細をすべて隠蔽して、それらのユーザが、移植性の高いコードを記述できるようにする。

6

【００１１】ＡＢＩに依存するものは、ＳＯＭ変数バッファ（somVaBuf）データ構造内でユーザから「隠蔽」される。新しい機能は、somVaBuf構造に含まれるva_listを操作する。これらの機能には、プラットフォーム特有のva_listを含むsomVaBufを作成し、somVaBuf構造内のva_listに引数を加え、somVaBuf構造からva_listを取り出してそれをＣライブラリー関数が作成したかのように使用し、somVaBufとそれに含まれるva_listを破壊し、va_list内の第１引数を調べて検討後に修正することができることが含まれる。ＡＰＩは、新しいva_listが現れるごとに、上記の機能を支援するライブラリーを追加する。その利点は、ユーザ・プログラムが変化せず、したがって移植性が高いことである。

【００１２】新しい機能は、次の通りである。

１．somVaBufを作成する。

```
void *somVaBuf_create(char *vb.int size)
```

入力：

vb　　ユーザに割り振られたメモリへのポインタまたはヌル

int　　ユーザ・メモリのサイズまたはヌル

出力：

somVaBufまたはヌル

【００１３】この機能は、somVaBuf構造を作成し初期設定する。この構造は、ユーザ可変引数を含むこともできる。

【００１４】２．va_listに引数を追加する。

```
long somVaBuf_add(somVaBuf vb,char *arg.int type)
```

入力：

vb　　somVaBuf(somVaBuf_createの結果)

arg　　va_listに追加する引数へのポインタ

type　　引数タイプ(TCKind)

出力：

成功　0

失敗　1

【００１５】この機能は、渡された引数をsomVaBuf構造に含まれるva_listに追加する。

【００１６】３．somVaBufからva_listを初期設定する。

```
void somVaBuf_get_valist(somVaBuf,va_list *ap)
```

入力：

vb　　somVaBuf構造へのポインタ

※ap　　va_listへのポインタ

出力：なし

【００１７】この機能は、somVaBuf構造内のva_listを入力引数として渡されたva_listに複写する。

【００１８】４．somVaBufのクリーンアップ。

```
void somVaBuf_destroy(somVaBuf vb)
```

入力：vb　　somVaBufデータ構造へのポインタ

出力：なし

【００１９】この機能は、somVaBuf構造及びsomVaBuf構造に含まれるva_listに関連する割り振り済みメモリを解放する。

【００２０】５．初期設定したva_list内の第１引数を取り出す。

```
unsigned long somvalistGetTarget(va_list ap)
```

入力：ap　　初期設定済みva_list

出力：va_list上の第１スカラ値

【００２１】この機能によって、ユーザは、初期設定済みのva_list内の第１スカラ値を調べることができる。

【００２２】６．初期設定済みva_list内の第１引数を

修正する。

void somvalistSetTarget(va_list ap,unsigned long val)

入力：

ap　初期設定済みva_list

val　va_list内の第１スカラ値に置き換わる値

出力：なし

【００２３】この機能により、ユーザはすでに初期設定済みのva_list内の第１引数を置き換えることができる。

【００２４】図３は、プラットフォーム特有のva_listを含むsomVaBuf機構を呼び出すためのコードの実施態様を示すブロック図である。Void *somVaBuf_create(char *vbuf, int size)。最初に機構は、ブロック３１０で、入力引数vbufがゼロかどうか判定する。ゼロでない場合は、ブロック３１２に進む。ゼロの場合、機構はブロック３１６に進み、そこでsomVaBufにスペースを割り振る。ブロック３１２で、システムは、入力引数sizeがゼロかどうか判定し、ゼロでない場合はブロック３１４に進む。sizeがゼロの場合は、システムはブロック３１６に進む。ブロック３１４において、システムは、sizeがVaBufのsizeよりも小さいかどうか判定し、小さい場合は、ブロック３１６に進み、小さくない場合は、システムはブロック３１８に進む。ブロック３１８において、システムは、VaBufをsomVaBufとして初期設定し、次にブロック３２０に進み、そこでシステムは、初期設定済みのsomVaBufを処理のために戻す。ブロック３１６の後も、同様に、システムはブロック３２０に進む。

【００２５】次に、図４と図５を参照すると、somVaBuf構造内のva_listに引数を加える機能が、図４と図５のブロック図に示されている。これは、Ｃ言語でlong somVaBuf_add(void* vbuf,char*arg,int type)の引数を有する。最初に、ブロック４１０で、システムは、somVaBuf内のva_listに加えられる引数のタイプを判定する。図４と図５において、引数が浮動小数点演算または二倍精度演算であるfloatとdoubleの場合、システムは右側に枝分かれし、character、boolean、octet、ushort、short、long、ulong、enumerator、stringまたはpointerなど残りの引数の場合は、左側に枝分かれする。まず、floatとdoubleの場合について説明する。最初に、ブロック４１２で、システムは、引数のタイプが浮動小数点かどうか判定し、そうでない場合は、ブロック４１４に進み、幅をdoublewideの引数として定義する。そうである場合、システムは、幅を浮動小数点引数として設定する。いずれの場合も、処理が完了すると、システムは、次の図５に続く分岐Ｂに進む。

【００２６】タイプが、floatとdouble以外の、前に述べたその他のタイプの場合、システムはブロック４１８～４２２に進み、そこで引数のタイプが、それぞれ、character、octet、boolean、short、u-short引数のどれであるかを判定する。ブロック４１８～４２２で、引数のタイプが、character、octetまたはboolean引数であると機構が判定した場合、システムはブロック４２８に進む。それぞれの検査後の更新を回避するため、機構は、各検査後に、ユーザ引数へのポインタでローカル変数wlongを初期設定する。検査がすべて完了すると、ブロック４３６でva_listを更新することができる。タイプがshortでもu-shortでもない場合、システムはブロック４３２に進み、そこで引数をu-longとして確立する。この時点で、システムは、図５の点Ａに進む。

【００２７】分岐Ａに沿って、機構はブロック４３４に進み、そこで機構は、va_list内で4バイト使用可能かどうか判定し、使用可能な場合は、ブロック４３６に進み、そうでない場合はシステムはブロック４３８に進む。ブロック４３６において、機構は、wlongが指すユーザ引数をva_listに入れてから主プログラムに戻る。ブロック４３８で、システムは、size-alignを４バイトにセットし、ブロック４４０～４４４に進んでから戻る。ブロック４４０で、機構は、va_listに十分なメモリを割り振る。ブロック４４２で、機構は、somVaBuf情報ファイルを更新する。ブロック４４４で、機構は、位置合わせされた値をva_listに入れてから戻る。

【００２８】機構は、分岐Ｂに沿って、ブロック４４６で、va_list内で８バイトが使用可能かどうかを判定する。使用可能である場合、システムはブロック４４８に進み、そうでない場合は、機構はブロック４５０に進む。ブロック４４８において、機構は、size_alignを８バイト幅に設定し、その後、機構は、前述のブロック４４０～４４４に進む。ブロック４５０において、機構は、widened引数をva_listに入れてから戻る。

【００２９】図６は、somVaBuf構造からva_list構造を取り出す機能の流れを示すブロック図である。これは、Ｃ言語でvoid somVaBuf_valist(void*vbuf,va_list*va)として実施される。ブロック５１０で、機構は、somVaBufからva_listを取り出す。

【００３０】次に、図７は、somVaBuf構造とそれに含まれるva_listを破壊する機能の実施態様を表す流れ図を示す。これは、ＣコードでsomVaBuf_destroy(void* vbuf)として実施される。最初に、ブロック６１０で、機構は、somVaBufにスペースが割り振られているかどうか判定し、割り振られていない場合は戻る。機構がスペースを割り振り済みである場合、システムはブロック６１２に進み、そこでメモリが解放される。次に、システムは戻る。図８は、va_list内の第１引数を構成後に調べる機能のブロック図である。これは、Ｃ言語ではunsigned long somva_listGetTarget(va_list ap)である。最初に、ブロック７１０で、機構が、入力引数apが指すva_list内の第１スカラを見つける。第１スカラが見つかったら、この値をさらに処理するために戻す。

【００３１】図９は、va_list内の第１引数を構成後に

修正する機能のブロック図である。これは、C言語でvoid somva_listSetTarget(va_list ap,unsigned long val)として実施される。最初の段階すなわちブロック810で、第1低スカラの記憶場所を決定する。この記憶場所の決定後、その記憶場所に値を入れる。

【0032】本発明を好ましい実施形態に関して具体的に示し説明してきたが、当業者は、本発明の趣旨及び範囲から逸脱せずに形態及び細部に様々な変更を行うことができることを理解されよう。

【0033】まとめとして、本発明の構成に関して以下の事項を開示する。

【0034】（1）1つまたは複数のプロセッサと、記憶システムと、1つまたは複数の入出力制御部と、プロセッサ、記憶システム及び入出力制御装置を接続するシステム・バスと、情報処理システムの動作を制御するようにプログラムされたオペレーティング・システムと、オペレーティング・システム・プログラムと連動して動作し、変数リストを操作するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェース（ＡＰＩ）プログラムとを含む情報処理システム。

（2）アプリケーション・プログラミング・インターフェース（ＡＰＩ）プログラムが、さらに、ユーザの可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する手段を含むことを特徴とする上記（1）に記載のシステム。

（3）ＡＰＩプログラムがさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を加える手段を含むことを特徴とする上記（2）に記載のシステム。

（4）ＡＰＩプログラムがさらに、前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する手段を含むことを特徴とする上記（3）に記載のシステム。

（5）ＡＰＩプログラムがさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する手段を含むことを特徴とする上記（3）に記載のシステム。

（6）前記ＡＰＩがさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラ値を取り出す手段を含むことを特徴とする上記（3）に記載のシステム。

（7）前記ＡＰＩがさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに、第1引数を入れることを可能にする手段を含むことを特徴とする上記（3）に記載のシステム。

（8）変数リストを操作して、アプリケーション・バイナリ・インターフェースを所与のプラットフォームから1つまたは複数の追加プラットフォームに移植するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェース（ＡＰＩ）を提供する手段を含むコンピュータ読取り媒体。

（9）アプリケーション・プログラミング・インターフェースがさらに、ユーザ可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する手段を含むことを特徴とする上記（8）に記載のコンピュータ読取り媒体。

（10）ＡＰＩ手段がさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を追加する手段を含むことを特徴とする上記（9）に記載のコンピュータ読取り媒体。

（11）ＡＰＩ手段がさらに、前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する手段を含むことを特徴とする上記（9）に記載のコンピュータ読取り媒体。

（12）ＡＰＩ手段がさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する手段を含むことを特徴とする上記（8）に記載のコンピュータ読取り媒体。

（13）ＡＰＩ手段がさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラ値を取り出す手段を含むことを特徴とする上記（8）に記載のコンピュータ読取り媒体。

（14）ＡＰＩ手段がさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに、第1引数を入れることを可能にする手段を含むことを特徴とする上記（8）に記載のコンピュータ読取り媒体。

（15）アプリケーション・バイナリ・インターフェース（ＡＢＩ）を、情報処理システム上で動作する選択されたプラットフォームから1つまたは複数の追加プラットフォームに移植する方法であって、変数リストを操作して、前記ＡＢＩを前記選択されたプラットフォームから前記1つまたは複数の追加プラットフォームに移植するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェースを提供する段階を含む方法。

（16）前記アプリケーション・プログラミング・インターフェース（ＡＰＩ）がさらに、ユーザ可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する段階を含むことを特徴とする上記（15）に記載のＡＢＩを移植する方法。

（17）前記ＡＰＩがさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を追加する段階を含むことを特徴とする上記（16）に記載のＡＢＩを移植する方法。

（18）前記ＡＰＩがさらに、前記オブジェクト変数バ

11

ッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する段階を含むことを特徴とする上記（１６）に記載のＡＢＩを移植する方法。

（１９）前記ＡＰＩがさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する段階を含むことを特徴とする上記（１６）に記載のＡＢＩを移植する方法。

（２０）前記ＡＰＩがさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第１スカラ値を取り出す段階を含むことを特徴とする上記（１６）に記載のＡＢＩを移植する方法。

（２１）前記ＡＰＩがさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リスト内に最初の引数を入れることを可能にする段階を含むことを特徴とする上記（１６）に記載のＡＢＩを移植する方法。

【図面の簡単な説明】

【図１】本発明による方法を実行するためのシステムのブロック図である。

【図２】本発明を支援するオペレーティング・システム・プラットフォーム及びシステム・オブジェクト・モデル・プログラムを示すブロック図である。

【図３】somVaBuf機構を呼び出すためのコードの実施態様を示すブロック図である。

12

【図４】somVaBuf構造内のva_listに引数を加える機能を示す流れ図である。

【図５】somVaBuf構造内のva_listに引数を加える機能を示す流れ図である。

【図６】somVaBuf構造からva_list構造を取り出す機能を示す流れ図である。

【図７】somVaBuf構造とそれに含まれるva_listを破壊する機能の実施態様を示す流れ図である。

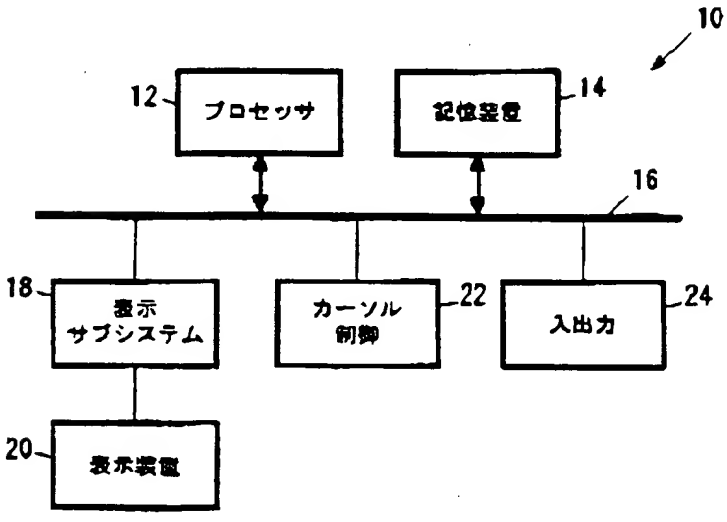
【図８】構成後にva_list内の第１引数を検査する機能のブロック図である。

【図９】構成後にva_list内の第１引数を修正する機能のブロック図である。

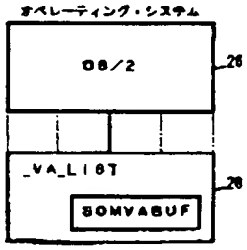
【符号の説明】

１０　情報処理システム
１２　高性能プロセッサ
１４　記憶システム
１６　システム・バス
１８　表示サブシステム
２０　表示装置
２２　カーソル制御装置
２４　入出力制御装置
２６　マルチタスク・オペレーティング・システム

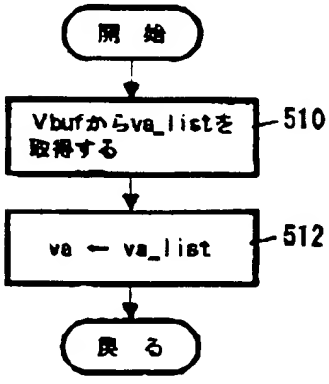
【図１】



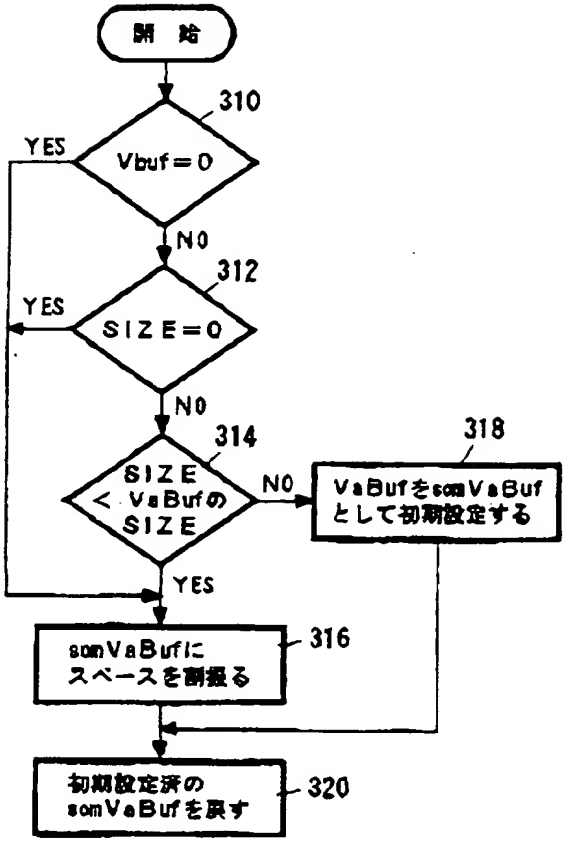
【図２】



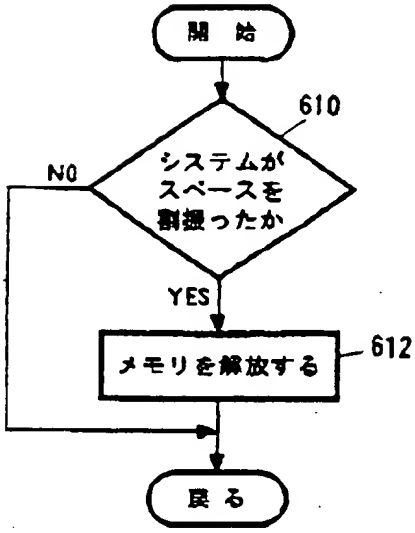
【図６】



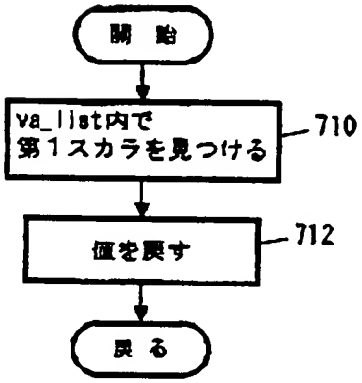
【図3】


開 始
310
Vbuf=0
YES
NO
312
SIZE=0
YES
NO
314
SIZE < VaBufのSIZE
NO
318
VaBufをsomVaBufとして初期設定する
YES
somVaBufにスペースを割振る
316
初期設定済のsomVaBufを戻す
320

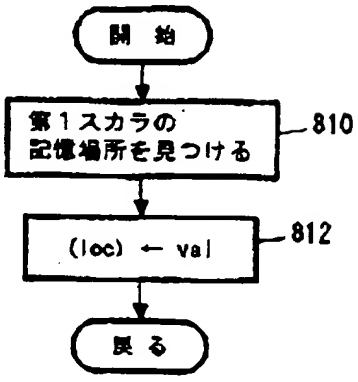
【図7】


開 始
610
システムがスペースを割振ったか
NO
YES
メモリを解放する
612
戻 る

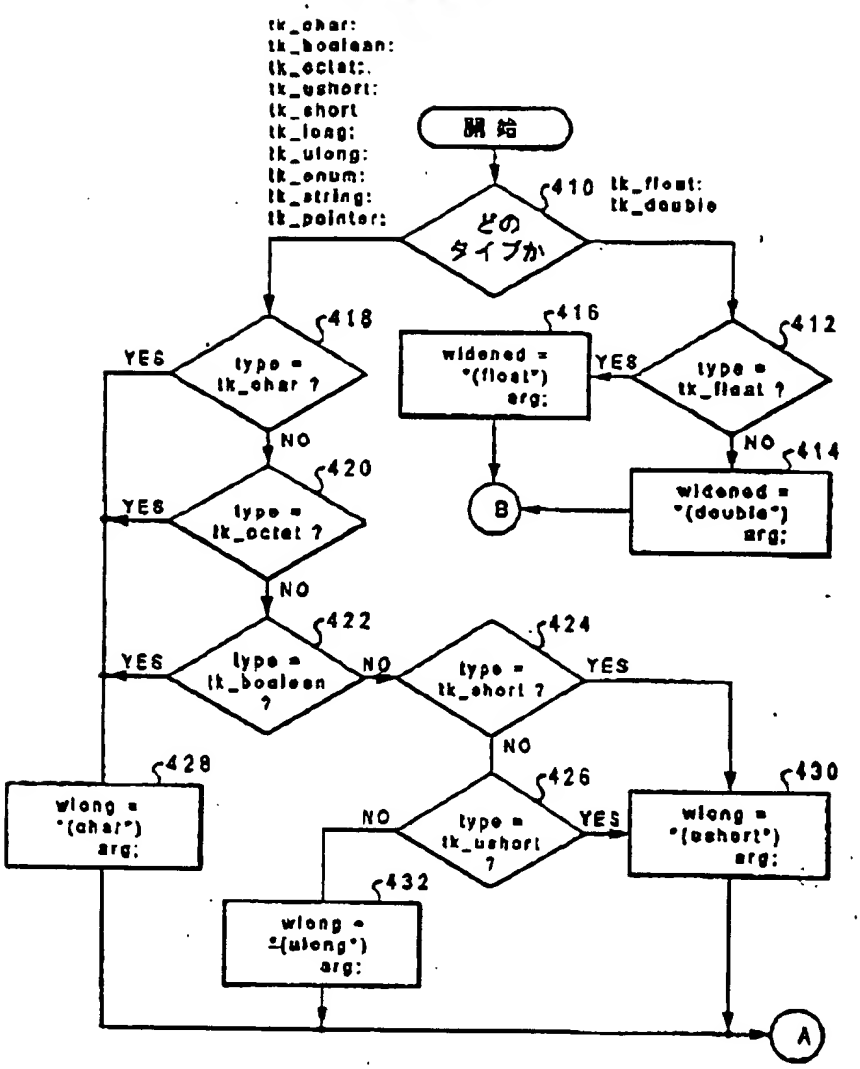
【図8】


開 始
va_list内で第1スカラを見つける
710
値を戻す
712
戻 る

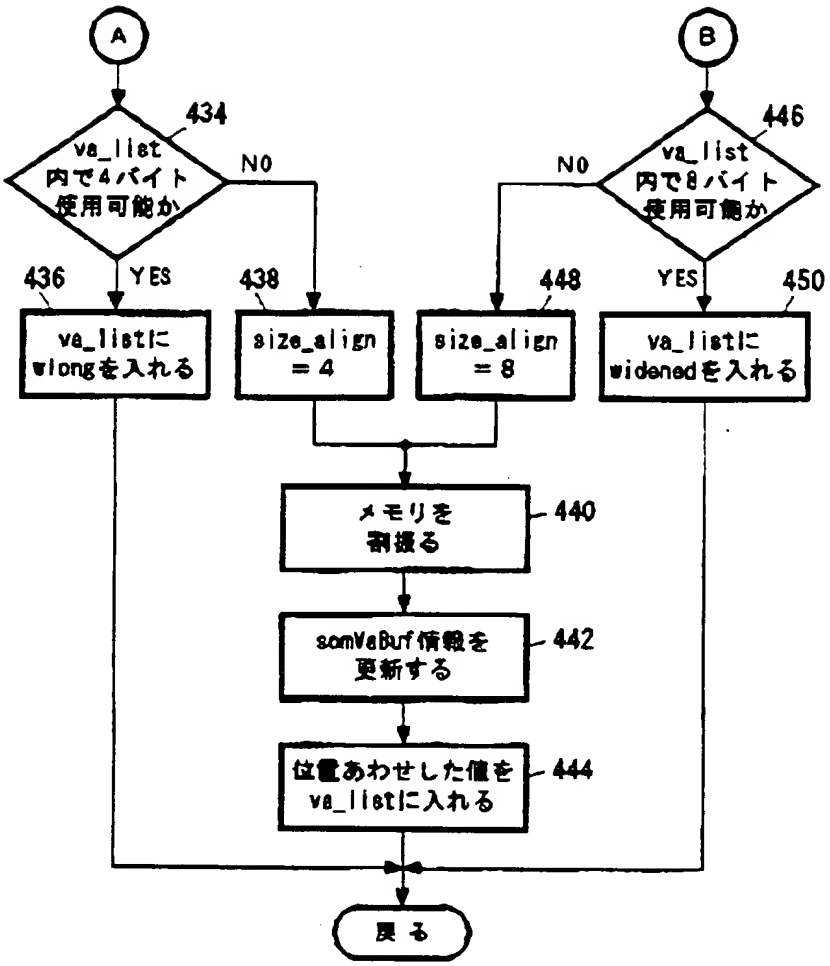
【図9】


開 始
第1スカラの記憶場所を見つける
810
(loc) ← val
812
戻 る

【図4】


tk_char:
tk_boolean:
tk_octet:
tk_ushort:
tk_short
tk_long:
tk_ulong:
tk_enum:
tk_string:
tk_pointer:
開始
410
どのタイプか
tk_float:
tk_double
418
type = tk_char ?
YES
NO
416
widened = *(float*) arg;
412
type = tk_float ?
YES
NO
414
widened = *(double*) arg;
B
420
type = tk_octet ?
YES
NO
422
type = tk_boolean ?
YES
NO
424
type = tk_short ?
YES
NO
428
wlong = *(char*) arg;
426
type = tk_ushort ?
NO
YES
430
wlong = *(ushort*) arg;
432
wlong = *(ulong*) arg;
A

【図5】



フロントページの続き

(72)発明者 プラカシュ・ヴィノドライ・デサイ
アメリカ合衆国78750 テキサス州ラウン
ド・ロック ロビン・トレール 1202